2_Week_4 | Project

BACK

I want you to create an Express application that consist of:

- a login/signup system, with an authentication based on JWT Token
- a route that return the list of users of the app (need to be authenticated)
- **new!** a route that return a specific user of the app (need to be authenticated)
- **new!** a route that edit the current user of the app (need to be authenticated)
- a route that return the list of messages of the current users (need to be authenticate)
- a route that send a message to a specific user (need to be authenticated)
- a route that update a message (only the *reader* can use this route and can update only `read` and `readDate`)
- **new!** a route that create a product and attach it to the current user (need to be authenticated)
- **new!** a route that return the list of products (need to be authenticated)
- **new!** a route that return the list of a specific product (need to be authenticated)
- **new!** a route that return the list of products of the current users (need to be authenticated)

The different routes of the backend should be **exactly as** the description in the next page.

FRONT

I want you to create a Vue application that consist of two view:

- a (non logged) view that contain:
    - a login component
    - a signup component

  (only one can be displayed at the same time)

- a (logged) view that contain:
    - HOME: **new!** a view that display the list of the last ten products
    - USERS: a view that display the users list. In the users list, you should have a 'Send Message' button for each users
        - when clicking on 'Send Message' button, you should go to a view that can send a message to the specific user
    - MESSAGES: a view that display the list of messages of the current user. Each message can have two state : read / unread. Each of this state should be represented through a different style. In this list, a message should only show the sender and the title of the message.
        - when clicking on a message, we should go to another view that display the full detail of the message. After this action (clicking on a message), we should update the message in the BE and update his read and readDate value.
    - **new!** PROFILE: a view that display the current user info. This view should contain a menu where you can navigate through three sub view:
        - a view that contain a form where you can change current user info (everything can be changed, except the username)
        - a view that contain the list of products of the current user
        - a view that contain a form to create a new product

— —

We need to navigate between the main views through a menu.

For the message lists view, you can inspire yourself with gmail.

You need to implement **Vue-router.**

You need to connect every component with his corresponding routes in BE (using Axios or Vue-ressource)

All datas should be stored in a database.

BONUS

Front:

- in the home view, display a map using Leaflet
- geolocate the current user, and add external api to retrieve exact address
- display in the map the ten last products

BACK END DESCRIPTION

LOGIN

Url: '**POST** /auth/login'
Parameters (body):
```
{
        username: String,
        password: String
}
```
Return value:
```
{
        success: Boolean,
        token: JWT Token,
}
```

SIGNUP

Url: '**POST** /auth/signup'
Parameters (body):
```
{
        username: String,
        password: String,
        firstName: String (optional),
        lastName: String (optional)
}
```
Return value:
```
{
        success: Boolean,
        content: User Profile (User: see below)
}
```

USERS

Url: '**GET** /users' | *Need authentication*
Return value:
```
{
        success: Boolean,
        content: List of Users (User: see below {without password})
}
```
Url: '**GET** /users/:id' | *Need authentication*
Return value:
```
{
        success: Boolean,
        content: User (User: see below {without password})
}
```

MESSAGES

Url: '**GET** /messages' | *Need authentication*
Return value:
```
{
        success: Boolean,
        content: List of Messages of the current user (Messages: see below)
}
```

Url: '**POST** /messages' | *Need authentication*
Parameters (body):
```
{
        title: String,
        content: String,
        userID: ID or username of the User who will received the message
}
```
Return value:
```
{
        success: Boolean,
}
```
Url: '**PUT** /messages/:messageID' | *Need authentication*
Parameters (body):
```
{
        read: Booelan,
        readDate: Date of read
    }
Return value:
    {
        success: Boolean,
    }
```

PRODUCTS

Url: ’**GET** /products’ | *Need authentication*
Return value:

```
    {
        success: Boolean,
        content: List of Products (User: see below {without password})
    }
```

Url: ’**GET** /products/:id’ | *Need authentication*
Return value:

```
    {
        success: Boolean,
        content: Product (User: see below {without password})
    }
```

Url: ’**POST** /products’ | *Need authentication*
Parameters (body): **Product Object (except id, userId and address)** (see below)
Return value:

```
    {
        success: Boolean,
    }
```

PROFILE

Url: ’**GET** /profile’ | *Need authentication*
Return value:

```
    {
        success: Boolean,
        content: Current connected user (User: see below {without password})
    }
```

Url: ’**GET** /profile/products’ | *Need authentication*
Return value:

```
    {
        success: Boolean,
        content: List of products of the current user (Products: see below)
    }
```

Url: ’**PUT** /profile’ | *Need authentication*
Parameters (body):

```
    {
        … every field of the User object (except the username and id)
    }
```

Return value:

```
    {
        success: Boolean,
    }
```

DATA STRUCTURE

User is represented as:
```
{
        id: Uniq User ID,
        username: String, | should be an email, and should be uniq,
        password: String,
        firstName: String (optional),
        lastName: String (optional),
        address: Address Object (see below, optional)
        creationDate: Date
}
```

Message is represented as:
```
{
        id: Uniq Message ID,
        title: String,
        content: String,
        creationDate: Date,
        read: Boolean,
        readDate: Date (optional),
        senderId: ID or username of the User who sent the message,
        receiverId: ID or username of the User who received the message
}
```

Product is represented as:
```
{
        id: Uniq Message ID,
        creationDate: Date Object,
        title: String,
        description: String,
        price: String,
        pictures: List of pictures (optional)
        userId: ID or username of the User who created the product
        address: Address Object (see below, optional),
}
```

Address is represented as:
```
{
        country: String (optional),
        region: String (optional),
        city: String (optional),
        street: String (optional),
        longitude: Number (optional),
        latitude: Number (optional),
}
```

ERROR HANDLING

If someone try to access a route that need authentication but doesn't have a good token, it should automatically return this response:

```
        {
                success: false,
                message: 'You should be authenticated to access this route!'
```

```
}
```